PATENT
Docket No. **VIV/0013.02**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Arthur Rothstein	Examiner: Hoffman, Brandon S.
Serial No.: 10/711,491	Art Unit: 2136
Filed: September 21, 2004	APPEAL BRIEF
For: System Providing Methodology for Securing Interfaces of Executable Programs	

Mail Stop Appeal
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

BRIEF ON BEHALF OF ARTHUR ROTHSTEIN

This is an appeal from the Final Rejection mailed April 19, 2007, in which currently-pending claims 1-47 and 49-60 stand finally rejected. Appellant filed a Notice of Appeal on July 19, 2007. This brief is submitted electronically in support of Appellant's appeal.

TABLE OF CONTENTS

1. REAL PARTY IN INTEREST

The real party in interest is assignee Check Point Software Technologies, Inc., located at 800 Bridge Parkway, Redwood City, CA 94065.

2. RELATED APPEALS AND INTERFERENCES

There are no appeals or interferences known to Appellant, the Appellant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

3. STATUS OF CLAIMS

The status of all claims in the proceeding is as follows:

Rejected: Claims 1-47 and 49-60

Allowed or Confirmed: None

Withdrawn: None

Objected to: None

Canceled: Claim 48

Identification of Claims that are being appealed: Claims 1-47 and 49-60

An appendix setting forth the claims involved in the appeal is included as Section 8 of this Appeal Brief.

4. STATUS OF AMENDMENTS

One Amendment has been filed in this case. Appellant filed an Amendment on February 2, 2007, in response to a non-final Office Action dated October 3, 2006. In the Amendment, the pending claims were amended in a manner which Appellant believes clearly distinguished the claimed invention over the art of record, for overcoming the art rejections. In response to the Examiner's Final Rejection dated April 19, 2007 (hereinafter "Final Rejection") finally rejecting Appellant's claims, Appellant filed a Notice of Appeal. Appellant has chosen to forgo filing an Amendment After Final that

might further limit Appellant's claims, as it is believed that further amendments to the claims are not warranted in view of the art. Accordingly, no Amendments have been entered in this case after the date of the Final Rejection.

5. SUMMARY OF CLAIMED SUBJECT MATTER

A. First Ground

As to Appellant's **First Ground** for appeal, where Appellant asserts that the Examiner's rejection of claim 31 under 35 U.S.C. Section 101 on the basis of non-statutory subject matter is improper, where the claimed invention is set forth in the embodiment in **independent claim 31**: A system for securing a program comprised of a plurality of interoperable components (see generally, Appellant's specification at paragraphs [13] -[14], paragraph [21], paragraph [57]; Fig. 4; Fig. 5), the system comprising: a module for extracting export information about a function of a first component of the program that is callable by at least one other component of the program (Appellant's specification, paragraph [21], paragraph [65], Abstract; Fig. 4 at 450; Fig. 5 at 550; Fig. 6 at 605 (remove exports from export table of module to be secured)); a module for securing the extracted export information (Appellant's specification, paragraph [21], paragraphs [65] -[67], paragraph [83] ; Fig. 4 at 450, 460; Fig. 5 at 550, 560; Fig. 6 at 607 (secure export table built to contain secured exports)); a validation module for validating authenticity of a second component attempting to obtain export information to invoke the function of the first component (Appellant's specification, paragraph [21], paragraph [69], paragraphs [72]-[74]; Fig. 4 at 440, 450; Fig. 5 at 540, 550; Fig. 7 at 701-709); and a security module for blocking the attempt to invoke the function of the first component if the second component cannot be authenticated (Appellant's specification, paragraph [21], paragraphs [74]-[75], paragraph [78] paragraph [94], Fig. 4 at 450; Fig. 5 at 550).

B. Second Ground

As to Appellant's **Second Ground** for appeal, Appellant asserts that the art rejection under **Section 103(a)** relying on **Bodrov** (U.S. Patent No. 6,802,006) in view of **Ferguson** (U.S. Patent No. 5,993,826) fails to teach or suggest all of the claim limitations

of Appellant's claimed invention, where the claimed invention is set forth in the embodiment in **independent claim 1:** A method for securing a program comprised of a plurality of interoperable components (see generally, Appellant's specification at paragraphs [13] -[14], paragraphs [19]-[20], paragraph [57]; Fig. 4; Fig. 5), the method comprising: extracting export information about a function of a first component of the program that is callable by at least one other component of the program (Appellant's specification, paragraphs [19]-[20], paragraph [65], Abstract; Fig. 4 at 450; Fig. 5 at 550; Fig. 6 at 605 (remove exports from export table of module to be secured)); securing the extracted export information (Appellant's specification, paragraphs [19]-[20], paragraphs [65] -[67], paragraph [83]; Fig. 4 at 450, 460; Fig. 5 at 550, 560; Fig. 6 at 607 (secure export table built to contain secured exports)); in response to an attempt by a second component to invoke the function of the first component, validating authenticity of the second component (Appellant's specification, paragraphs [19]-[20], paragraph [69], paragraphs [72]-[74]; Fig. 4 at 440, 450; Fig. 5 at 540, 550; Fig. 7 at 701-709); if the authenticity of the second component is validated, providing access to the function of the first component using the secured extracted export information (Appellant's specification, paragraphs [19]-[20], paragraphs [74]-[76], paragraph [83]; Fig. 4 at 450, 460; Fig. 5 at 550, 560); and otherwise, blocking the attempt by the second component to invoke the function (Appellant's specification, paragraphs [19]-[20], paragraphs [74]-[76], paragraph [94], paragraph [101]).

Appellant further asserts that the art rejection relying **Bodrov** and **Ferguson** fails to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention is set forth in the embodiment in **independent claim 15:** A method for securing a program comprised of a plurality of modules (see generally, Appellant's specification at paragraphs [13] -[14], paragraphs [19]-[20], paragraph [57]; Fig. 4; Fig. 5); at least one of the modules having export information for allowing other modules to invoke its program code (Appellant's specification at paragraph [20], paragraph [57], paragraph [62], paragraphs [65]-[67]; Fig. 4 at 420; Fig. 5 at 520), the method comprising: generating signatures for at least some of the program's modules (Appellant's specification at paragraph [20], paragraph [57], paragraph [69], paragraph [74], paragraphs [115]-[116]; as the program is loaded, validating said signatures so as to

verify authenticity of respective modules of the program (Appellant's specification at paragraph [20], paragraph [69], paragraph [72], paragraph [74]; Fig. 7 at 702, 705, 709); for each module having program code that may be invoked by another module, removing that module's export information (Appellant's specification, paragraphs [19]-[20], paragraph [65], Abstract; Fig. 4 at 450; Fig. 5 at 550; Fig. 6 at 605 (remove exports from export table of module to be secured)); securely storing any removed export information (Appellant's specification, paragraphs [19]-[20], paragraphs [65] -[67], paragraph [83]; Fig. 4 at 450, 460; Fig. 5 at 550, 560; Fig. 6 at 607 (secure export table built to contain secured exports)); for each module having its export information removed, blocking any attempt from another module to invoke its program code if the other module cannot be authenticated (Appellant's specification, paragraphs [19]-[20], paragraphs [74]-[76], paragraph [94], paragraph [101]); and if the other module is authenticated, allowing the attempt to proceed using the securely stored export information (Appellant's specification, paragraphs [19]-[20], paragraphs [74]-[76], paragraph [83]; Fig. 4 at 450, 460; Fig. 5 at 550, 560).

Appellant further asserts that the art rejection relying **Bodrov** and **Ferguson** fails to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention is set forth in the embodiment in **independent claim 31** (the mapping of which is shown above under Appellant's First Ground for appeal, and which hereby is incorporated by reference).

Appellant further asserts that the art rejection relying **Bodrov** and **Ferguson** fails to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention is set forth in the embodiment in **independent claim 45:** A method for securing an exported function of a program (Appellant's specification, paragraph [22], paragraphs [65]-[67], paragraph [72], paragraph [81], paragraphs [99]-[101]; Fig. 4; Fig. 5) the method comprising: extracting export information about the exported function of the program (Appellant's specification, paragraph [22], paragraph [65], Abstract; Fig. 4 at 450; Fig. 5 at 550; Fig. 6 at 605 (remove exports from export table of module to be secured)); securing the extracted export information (Appellant's specification, paragraph [22], paragraphs [65] -[67], paragraph [83]; Fig. 4 at 450, 460; Fig. 5 at 550, 560; Fig. 6 at 607 (secure export table built to contain secured exports)); intercepting an attempt to

access the exported function by an importer (Appellant's specification, paragraph [22], paragraphs [74] -[75], paragraph [78], paragraph [82], paragraph [101], Fig. 4 at 440, 450; Fig. 5 at 540, 550); authenticating the importer for determining whether to permit access to the exported function (Appellant's specification, paragraph [22], paragraph [69], paragraphs [72]-[74]; Fig. 4 at 440, 450; Fig. 5 at 540, 550; Fig. 7 at 709); if the importer is authenticated, providing access to the exported function based on the secured extracted export information (Appellant's specification, paragraph [22], paragraphs [74]-[76], paragraph [83]; Fig. 4 at 450, 460; Fig. 5 at 550, 560); and otherwise, blocking access to the exported function (Appellant's specification, paragraphs [19]-[20], paragraph [22], paragraphs [74]-[76], paragraph [94], paragraph [101]).

C. Third Ground

As to Appellant's **Third Ground** for appeal, Appellant asserts that the art rejection under **Section 103(a)** relying on **Bodrov** (U.S. Patent No. 6,802,006) in view of **Ferguson** (U.S. Patent No. 5,993,826), further in view of **Idoni** (U.S. Published Application No. 2004/0123308) fails to teach or suggest all of the claim limitations of Appellant's claimed invention**,** where the claimed invention is set forth in the embodiment in **independent claims 15, 31 and 46** (the mapping of which are shown above under Appellant's **First Ground** and **Second Ground** for appeal, and which hereby are incorporated by reference).

6. GROUNDS OF REJECTION TO BE REVIEWED

The grounds presented on appeal are:

(1st) Whether **claim 31** is unpatentable under 35 U.S.C. Section 101 as directed to non-statutory subject matter; and

(2nd) Whether **claims 1-19, 21-42, 45 and 49-60** are unpatentable under 35 U.S.C. 103(a) as being obvious over U.S. Patent No. 6,802,006 to Bodrov (referred to herein as "Bodrov" herein) in view of U.S. Patent No. 5,993,826 issued to Ferguson (referred to herein as "Ferguson").

(3rd) Whether **claims 20, 43, 44, 46 and 47** are unpatentable under 35 U.S.C. 103(a) as being obvious over Bodrov in view of Ferguson, further in view of U.S.

Published Application No. 2004/0123308 to Idoni (referred to herein as "Idoni").

7. ARGUMENT

A. First Ground: Claim 31 rejected under 35 U.S.C. Section 101

Claim 31 stands rejected under 35 U.S.C. Section 101 as directed to non-statutory subject matter. The Examiner states that Appellant's each module in Appellant's claim 31 would reasonable interpreted as software routines, which is a system of software per se and lacks the necessary physical articles or objects as components to make it a machine or manufacture.

Article 1, Section VIII of the U.S. Constitution authorizes patent protection for inventions that promote the progress of the useful arts. Congress acted upon this authorization in 35 U.S.C. Section 101 by mandating patent protection for inventions without regard to the form or physical embodiment of the invention as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.
>
> (35 U.S.C. Section 101)

Congress intended that "anything under the sun that is made by man" should be patentable, with the exception of "laws of nature, physical phenomena and abstract ideas." Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980); Diamond v. Diehr, 450 U.S. 175, 182 (1981). As noted in *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992), the Examiner bears the initial burden of presenting a *prima facie* case of unpatentability under Section 101. The Examiner has failed to meet this burden as Appellant's claimed invention is directed to statutory subject matter.

At the outset, Appellant's claims clearly fall within one or more of the statutory categories set forth in Section 101 as they comprise a "new and useful process, machine, manufacture or composition of matter" (or a new and useful improvement thereof). Thus, as Appellant respectfully believes that the claimed invention falls within the statutory categories of invention set forth in Section 101, they are patentable unless they comprise

"laws of nature, physical phenomena and abstract ideas." In addition, while abstract ideas, natural phenomena, and laws of nature are not eligible for patenting, methods and systems employing abstract ideas, natural phenomena, and laws of nature to perform a real-world function may be patentable. Appellant does not agree that software falls outside of Section 101 as not comprising a new and useful process, machine, manufacture, or composition of matter under the standards described above.

In addition, Appellant respectfully believes that the Examiner has incorrectly construed Appellant's specification and claims as indicating that the elements of Appellant's invention are implemented solely in software. Appellant's specification expressly states that the elements of the invention may be implemented in hardware, software or firmware or combinations thereof. This is expressly stated, for example, in Appellant's specification as follows: "...the corresponding apparatus element may be configured in hardware, software, firmware or combinations thereof" (Appellant's specification, paragraph [43], emphasis added). Appellant's specification also describes in detail a computer hardware and software environment in which Appellant's invention may be implemented (Appellant's specification, paragraphs [44]-[55]). Moreover, Appellant states that the software (computer-executable instructions) direct operation of a device under processor control, such as a computer (Appellant's specification, paragraph [108]). As Appellant's claimed invention comprises a hardware and software combination, Appellant respectfully believes that it comprises statutory subject matter.

For the reasons set forth above, it is respectfully submitted that the Examiner's rejection of Claim 31 under 35 U.S.C. Section 101 as directed to non-statutory subject matter should be not be sustained.

B. Second Ground: Claims 1-19, 21-42, 45 and 49-60 rejected under 35 U.S.C. Section 103

1. General

Under Section 103(a), a patent may not be obtained if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. To establish a prima facie

case of obviousness under this section, the Examiner must establish: (1) that there is some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings, (2) that there is a reasonable expectation of success, and (3) that the prior art reference (or references when combined) must teach or suggest all the claim limitations. (See e.g., MPEP 2142). The reference(s) cited by the Examiner fail to meet these conditions.

2. Claims 1-19, 21-42, 45, and 49-60

Claims 1-19, 21-42, 45, and 49-60 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,802,006 to Bodrov ("Bodrov") in view of U.S. Patent No. 5,993,826 issued to Ferguson ("Ferguson"). The following rejection of Appellant's claims 1, 12, 14, 29-31, 45, 59, and 60 by the Examiner is representative of the Examiner's rejection of the Appellant's claims as being unpatentable over Bodrov and Ferguson:

> Regarding claims 1, 12, 14, 29-31, 45, 59, and 60, Bodrov teaches a method/system/computer-readable medium for securing a program comprised of a plurality of interoperable components, the method comprising:
>
> Extracting export information about a function of a first component of the program that is callable by at least one other component of the program (fig. 3, ref. num 321 and col. 5, lines 14-18);
>
> Securing the extracted export information (fig. 5);
>
> In response to an attempt by a second component to invoke the function of the first component, validating authenticity of the second component (fig. 6);
>
> If the authenticity of the second component is validated, providing access to the function of the first component using the secured extracted information (fig. 6, ref. num 715); and
>
> Otherwise, blocking the attempt by the second component to invoke the function (fig. 6, ref. num 714).
>
> Bodrov does not specifically teach securing the extracted information, however, Bodrov does teach extracted export information.
>
> Ferguson teaches securing extracted information (col. 8, lines 38-48), which has

been extracted as export information by Bodrov.

> It would have been obvious to one of ordinary skill in the art, at the time the invention was made, to combine securing extracted information, as taught by Ferguson, with the method of Bodrov. It would have been obvious for such modifications because securing the extracted information prevents access to the information unless permission has been given (see col. 8, lines 38-48 of Ferguson).

(Final Rejection, paragraph 7)

Here the Examiner relies substantially on the Bodrov reference in the rejection of Appellant's claims by equating Bodrov's system for verifying authenticity of dynamically connectible executable images with Appellant's system and methodology for securing interfaces of executable files. At the outset, Appellant does not claim to have invented the notion of validating authenticity of a file. Although at a high level both Bodrov's system and Appellant's invention validate authenticity as part of their operations, Appellant's claimed invention is focused on securing internal interfaces of executable files and includes specific elements that distinguish it from Bodrov, Ferguson and the other prior art of record.

Appellant's invention addresses a specific security vulnerability of programs comprised of a plurality of interoperable components in which a hacker or other perpetrator seeking to attack a program can use export information (e.g., in export tables of a component) to identify a function that can be exploited in an attack and to determine the location of the function (Appellant's specification, paragraph [0062]). Appellant's invention provides for securing this export information, so as to block attacks which attempt to obtain the function name, address or other such information for use in an attack on the program (Appellant's specification, paragraph [0067]). Thus, Appellant's claimed invention is focused on blocking attacks attempting to use **exported functions** (and associated export information) of a program consisting of interoperable components to attack the program. Bodrov's system, in contrast, focuses on examining **importers** (e.g., an importing module) and the pointers to external modules included in its import tables. These differences become clear when one compares the operations of Bodrov's system to Appellant's claimed invention.

Bodrov's system validates the authenticity of executable images by first

generating a reference digital signature for an executable image (Bodrov col. 6, lines 41-50; Fig. 5 at 608). This reference digital signature is then stored (e.g., stored on a storage device, appended to the executable image, etc.) for later retrieval (Bodrov col. 6, lines 51-60; Fig.5 at 612). Subsequently, when an executable image is loaded into memory, the validator of Bodrov's system generates an authenticity digital signature to ensure that the executable image has not been tampered with (Bodrov col. 7, lines 1-6; Fig. 5 at 614). If the reference digital signature does not match the authenticity digital signature, Bodrov's validator initiates a security alert (Bodrov, col. 7, lines 12-18; Fig. 5 at 618, 622). Significantly, Bodrov's validator examines the authenticity of an executable image based on generating a digital signature for the image loaded into memory and comparing it to a prior "reference" digital signature which was previously generated for the same image.

Bodrov's validation system also includes functionality for verifying that each of the import pointers in the executable image is bound to a proper location (Bodrov, col. 8, lines 4-7; Fig. 6 at 708). Here, each of the import pointers in the code (executable image) is examined to determine if such import pointer is bound to a location within the import table. If so, then Bodrov's validator proceeds to determine whether the binding locations of external pointers in the import table reside within one of the executable images identified by the import table (Bodrov, col. 8, lines 37-40; Fig. 6 at 722). If either of these tests fails, Bodrov's system proceeds to initiate a security alert (Bodrov col. 8, lines 11-14 and lines 53-57; Fig. 6 at 714). Thus, Bodrov's validation system serves to validate authenticity of a given executable image and also provides for examining **import** pointers **in import tables** of the given executable image. Thus, Bodrov is focused on examining an **importer** (i.e., importing image or module) and the external modules or images that are referenced in its import tables.

Appellant's invention, in contrast, is focused on hardening and securing internal interfaces of an **exporter** (e.g., an exported function of a component intended to be used by other program components) so as to prevent improper use of such exported function (e.g., by unauthorized importers). A specific security vulnerability of programs comprised of a plurality of interoperable components is that a hacker or other perpetrator seeking to attack a program can use export information to identify a function that can be exploited in an attack (Appellant's specification, paragraph [62]). Accordingly,

Appellant's invention provides for extracting and securing this export information, so as to block attacks which attempt to obtain the function name, address or other such information for use in an attack on the program (Appellant's specification, paragraph [67]). Significantly, Appellant's claimed invention provides for securing this export information about an exported function of a program component by **extracting (i.e., removing) this export information from an export table of the component** (Appellant's specification, paragraphs [66]-[67]). The extracted information is then placed in a secure export table and is only made available to importers that are successfully validated. In this manner, Appellant's invention operates at runtime when an exported function of a program is invoked to only allow invocation of such function if the authenticity of the importer is validated. This feature of extracting and securing export information is specifically described in Appellant's claims including, for instance, in the following limitations of Appellant's claim 1:

> A method for securing a program comprised of a plurality of interoperable components, the method comprising:
> extracting export information about a function of a first component of the program that is callable by at least one other component of the program;
> securing the extracted export information;
>
> (Appellant's claim 1, emphasis added)

The Examiner references Bodrov at Fig. 3, reference number 321 and col. 5, lines 14-18 for the referenced teachings of extracting export information about a function of a component that is callable by other components (Final Rejection, paragraph 7). Although Fig. 3 of Bodrov does depict an export table at 321, Bodrov makes no mention of extracting or removing information from this export table. Instead, the referenced text of Bodrov simply describes an export table as follows:

> The export table 321 identifies each of the procedures that are made publicly available by the executable image 100. It is noted that the executable image 100 can include other information such as debug information or other tables to assist with the loading and/or linking process.
>
> (Bodrov, col. 5, lines 14-18)
>
> As illustrated above, Bodrov describes an export table in very general terms and

does not teach extracting export information from the export table in the manner provided in Appellant's specification and claims. Thus, it does not provide teachings equivalent to Appellant's claimed invention which provides for extracting information from standard export tables in order to guard against misuse of such export information.

Appellant's invention further provides for securing the export information extracted from export tables and only providing such information for use by other components that are authenticated. Here, the Examiner acknowledges that Bodrov does not teach securing the export information and references Ferguson for these teachings. However, turning to Ferguson one finds that its teachings are distinguishable from Appellant's claimed invention in a number of respects.

Ferguson describes an approach for securing objects stored in a distributed directory. Ferguson's system comprises a plurality of interconnected nodes which access a distributed directory including a hierarchy of objects, each of which have associated attributes. The distributed directory is a synchronized hierarchical database which maintains and provides access to information across the network of interconnected nodes (Ferguson, col. 4, lines 19-25). Information on the distributed directory can be created, read, modified, and shared by nodes having access rights to the distributed directory (Ferguson, col. 4, lines 33-36). Access to objects in the distributed directory of Ferguson is controlled using an access control mechanism which provides physical security, login security and directory security for the distributed directory (Ferguson, col. 6, lines 46-49, Fig. 3). The physical security and login security measures described by Ferguson are conventional steps of maintaining physical security of computer systems and requiring a user name and password for login to such systems (Ferguson, col. 6, lines 55-64). Ferguson also describes that directory security is used after login security has been first verified to assign specified rights to the objects stored in the distributed directory, such as rights to read, write, create and erase such objects (Ferguson, col. 7, lines 17-30) and rights to access or manage objects and/or change their properties (Ferguson col. 7, lines 33-44). Thus, Ferguson describes mechanisms for securing objects in a distributed directory which do not appear at all analogous to Appellant's claimed invention.

Even assuming the distributed directory of Ferguson is somehow analogous to Appellant's invention for securing internal interfaces of a program, Ferguson does not

include the specific teachings of securing export information about an exported function of a program which has been extracted from an export table (Appellant's specification, paragraphs [66]-[67]). The Examiner references Ferguson at col. 8, lines 38-48 for the teachings of securing extracted export information; however, the referenced portion of Ferguson simply provides as follows:

> A further level of security can be achieved in optional step 84, wherein the program is encrypted using an encryption system. At step 85, the program is stored as a value in the target attribute of the target object. If the program was encrypted, this cipher text form of the program should be stored. Once stored as a value, the replication system of the distributed directory can then replicate the value to one or more nodes that access the directory. Furthermore, the access control mechanism of the distributed directory will prevent access to the value, whereby the program cannot be executed unless access has been permitted by the access control mechanism.
>
> (Ferguson, col. 8. lines 38-48, emphasis added)

As illustrated above, Ferguson provides for encrypting a program using an encryption system and an access control system that regulates whether or not a given user is allowed to decrypt and execute the program. This is not comparable to Appellant's claimed invention which operates to secure internal interfaces of a program which has been loaded and is being operated. Appellant's solution does not encrypt the entire program, but rather extracts and secures only export information extracted from export tables of program components. Appellant's approach of removing the export information from the export tables provides additional security as it makes it more difficult for an attacker to determine the function name and the address of the function for purposes of initiating an attack using this export information (Appellant's specification, paragraphs [0078] and [0098]). With Ferguson's system, in contrast, once the program was decrypted and placed into use, the program would still include export information (e.g., address of an exported function) which could be obtained by an attacker using well-known system tools and utilities so as to launch an attack. Ferguson simply describes conventional steps of encrypting a program and regulating access to and/or the ability to decrypt the program by users through an access control mechanism (Ferguson, col. 8, lines 49-62 and col. 9 line 64 to col. 10 line 6). Ferguson's security measures do not,

however, provide any protection to the program against attack once it is decrypted and placed into operation.

Appellant's invention operates to secure a program against attack as it operates by regulating access to the secured export information. As the program runs, Appellant's solution operates automatically to only permit properly authenticated program components to access and use export information in order to invoke a given function, while denying access and use of the secured export information by unauthorized components (e.g., by attackers seeking to exploit internal interfaces of the program to launch an attack). In response to attempt to invoke an exported function of a first component by a second (importer) component, Appellant's solution examines the second component which is attempting to invoke the exported function. Appellant's claimed invention validates the authenticity of the second (importer) component before permitting access and use of the extracted export information for purposes of invoking the exported function of the first component (Appellant's specification, paragraph [74]). These features are specifically described in Appellant's claims. For instance, Appellant's claim 1 includes the following claim limitations:

> in response to an attempt by a second component to invoke the function of the first component, validating authenticity of the second component;
> if the authenticity of the second component is validated, providing access to the function of the first component using the secured extracted export information; and
> otherwise, blocking the attempt by the second component to invoke the function.
>
> (Appellant's claim 1, emphasis added)

As illustrated above, Appellant's invention operates in response to an attempt to invoke an exported function as it operates at runtime by examining the importer component that is attempting to invoke the exported function. Before providing access to the importer using the secured extracted export information, Appellant's solution validates the authenticity of the importer. In this manner, external attempts to invoke the exported function so as to launch an attack are intercepted.

The Examiner references Bodrov at Fig. 6, reference numbers 714 and 715 for the corresponding teachings; however reference number 714 of Fig. 6 simply states

"INITIATE SECURITY ALERT", while reference number 715 simply states "END". Thus, the relevance of the referenced portions of Fig. 6 is, to say the least, rather unclear. Moreover, when one examines the balance of Fig. 6 and the text of the Bodrov reference, one finds that Bodrov's system is focused on examining import tables of an importer and the external modules or images that are referenced in such import tables. As described above in greater detail, Bodrov's system operates to examine pointers in import tables of an importer, so as determine if the destination addresses fall within the export section of the expected executable image (Bodrov, Fig. 6, ref. number 716, 718, 722). This is not Appellant's approach. Appellant's claimed invention operates to extract and secure export information so as to secure an exported function against improper use. As Bodrov's system does not extract export information from export tables and does not secure the extracted export information, it does not secure the program from an attacker seeking to use the export information to launch an attack on the program.

3. Conclusion

All told, Bodrov and Ferguson, even if combined, include no teachings of extracting export information from export tables of a program and securing the extracted export information so as to guard against improper use of such information. The combined references also do not provide a security solution which operates at runtime, as a program is operating, to check that attempts to invoke exported functions of the program are from authorized, authenticated components of the program (rather than from hackers or other unauthorized external sources) as provided in Appellant's claims. For the reasons stated above, it is respectfully submitted that Appellant's claims distinguish over the combined references and that the Examiner's rejection under Section 103 should not be sustained.

C. Third Ground: Claims 20, 43, 44, 46 and 47 rejected under 35 U.S.C. Section 103

Claims 20, 43, 44, 46 and 47 stand rejected under 35 U.S.C. Section 103(a) as being obvious over Bodrov in view of Ferguson, further in view of U.S. Published Application No. 2004/0123308 to Idoni ("Idoni"). Here, the Examiner relies on Bodrov (above) and Ferguson (above) for the basic rejection but adds Idoni for the proposition

that it teaches Appellant's claim limitation of initializing the security module before other modules and the security module inserting executable code during initialization so as to direct an attempt to invoke an exported function to the security module. The claims are believed to be allowable for at least the reasons stated above in Appellant's **Second Ground** of Appeal pertaining to Bodrov and Ferguson (which are hereby incorporated by reference herein).

As set forth above, to establish a prima facie case of obviousness under Section 103, the Examiner must establish (among other things) that the references when combined teach or suggest all the claim limitations. Bodrov and Ferguson do not teach Appellant's claim limitation of extracting export information from export tables of a program and securing such export information against unauthorized use, nor do they operate at runtime to check that attempts to invoke exported functions are from authorized, authenticated program components. Idoni contains no teaching or suggestion that remedies these deficiencies.

Further, the claims are believed to be allowable for the following additional reasons. Regarding Appellant's claim limitations (e.g., in claim 43) of inserting executable code into the second module during initialization so as to direct an attempt by the second module to invoke a function of the first module to the security module, the Examiner cites Idoni at paragraph [0031] and Fig. 5. At paragraph [0031], Idoni states:

> This process is illustrated in greater detail with reference to FIG. 5. During production, in a step 502, compilation and linking as defined for implicit linkage results in an application program with unresolved external references to entities in DLLs. That is, a set of import libraries specifying one or more DLLs required by the application program is statically linked with the application program. In a step 504, the application program is loaded and executed by the computer operating system as defined for explicit linkage, including the unresolved references provided by the import libraries specified in step 502. In a step 506, the executing application invokes the DLL loader routine for any necessary DLLs. Included in the call to the loader routine is an identification and physical location of the DLL. In a step 508, the executing application invokes the linker routine to resolve any external references. Included in the call to the linker program is a memory base address of the loaded DLL necessary to resolve external references. In a step 510, the application executable continues as normal invoking entities provided by the DLLs seamlessly.

The foregoing teaching from Idoni simply describes the loading of DLL's which

are referenced in import libraries of an application program. This is not the same as the feature set forth in Appellant's claim limitation. Appellant's claim 43, when read in conjunction with its independent claim makes it clear that executable code is inserted during initialization of the second module (import module) so as to direct an attempt by the second module (import module) to invoke the function of the first module (export module) to the security module. This is also clearly shown, for example, in Appellant's Fig. 4 which provides that stub 440 (i.e., executable code) is inserted into an import module 410 so as to direct an attempt by the import module 410 to invoke an exported function of export module 440 to the security module (i.e., first load DLL 450), all as depicted at Appellant's Fig. 4. Comparing Fig. 5 of Idoni, one finds that it describes conventional steps of calling a loader to load DLLs (Idoni, Fig. 5 at 506) and calling a linker to resolve external references (Idoni, Fig. 5 at 508). Moreover, Appellant's claim 44 provides that if the importer (second module) is authenticated, the executable code is modified so that the importer can thereafter directly invoke the function (i.e., without having to go through the security module). Thus, the teachings of Idoni are not comparable to those of Appellant's claimed invention.

It is believed that the claims that the Examiner has rejected under Section 103 distinguish over the combined references and are thus patentable. Accordingly, it is respectfully requested that the Examiner's rejection of the claims of this group not be sustained.

D. Conclusion

The present invention addresses security vulnerabilities of programs comprised of a plurality of interoperable components by securing export information that a hacker or other perpetrator seeking to attack a program can use to launch an attack on a program. It is respectfully submitted that the present invention, as set forth in the pending claims, sets forth a patentable advance over the art.

In view of the above, it is respectfully submitted that the Examiner's rejections under 35 U.S.C. Section 101 and 103 should not be sustained. If needed, Appellant's undersigned attorney can be reached at 925 465 0361. For the fee due for this Appeal Brief, please refer to the attached Fee Transmittal Sheet. This Appeal Brief is submitted electronically in support of Appellant's Appeal.

Respectfully submitted,

Date: September 19, 2007

/G. Mack Riddle/

G. Mack Riddle; Reg. No. 55,572
Attorney of Record

925 465 0361
925 465 8143 FAX

8. CLAIMS APPENDIX

1. A method for securing a program comprised of a plurality of interoperable components, the method comprising:

extracting export information about a function of a first component of the program that is callable by at least one other component of the program;

securing the extracted export information;

in response to an attempt by a second component to invoke the function of the first component, validating authenticity of the second component;

if the authenticity of the second component is validated, providing access to the function of the first component using the secured extracted export information; and

otherwise, blocking the attempt by the second component to invoke the function.

2. The method of claim 1, further comprising:

generating a signature for at least one other component of the program authorized to call the function of the first component, so as to enable authentication of said one other component.

3. The method of claim 2, wherein said step of generating a signature includes generating a selected one of an Authenticode signature and an MD5 message digest.

4. The method of claim 2, wherein said step of generating a signature includes generating a hash of said one other component and encrypting the hash with a private key.

5. The method of claim 4, wherein said validating step includes decrypting the hash with a public key and comparing the hash to a known value.

6. The method of claim 1, wherein said extracting step includes removing the export information from an export table of the first component.

7. The method of claim 1, wherein said extracting step includes removing the function name from an export table of the first component.

8. The method of claim 1, wherein said securing step includes obscuring the function name.

9. The method of claim 1, wherein said securing step includes creating a secure export table for securing the extracted export information.

10. The method of claim 1, wherein said providing step includes routing a call by the second component to the function of the first component.

11. The method of claim 1, wherein said providing step includes returning an address of the function of the first component to the second component.

12. The method of claim 1, wherein said extracting step includes extracting export information about a function of the first program specified by a user.

13. A computer-readable medium having processor-executable instructions for performing the method of claim 1.

14. The method of claim 1, further comprising:

providing a downloadable set of processor-executable instructions for performing the method of claim 1.

15. A method for securing a program comprised of a plurality of modules, at least one of the modules having export information for allowing other modules to invoke its program code, the method comprising:

generating signatures for at least some of the program's modules;

as the program is loaded, validating said signatures so as to verify authenticity of respective modules of the program;

for each module having program code that may be invoked by another module, removing that module's export information;

securely storing any removed export information;

for each module having its export information removed, blocking any attempt from another module to invoke its program code if the other module cannot be authenticated; and

if the other module is authenticated, allowing the attempt to proceed using the securely stored export information.

16. The method of claim 15, wherein said generating step includes generating a selected one of an Authenticode signature and an MD5 message digest.

17. The method of claim 15, wherein said generating step includes generating a hash of a module and encrypting the hash with a private key.

18. The method of claim 17, wherein said validating step includes decrypting the hash with a public key and comparing the hash to a known value.

19. The method of claim 15, further comprising:

providing a security module for validating authenticity of a module.

20. The method of claim 19, wherein the security module includes instructions causing the security module to be initialized before other modules of the program.

21. The method of claim 19, wherein an attempt to invoke a module having its export information removed is routed to the security module.

22. The method of claim 21, wherein the security module allows the attempt to proceed if the other module making the attempt is authenticated.

23. The method of claim 15, wherein said allowing step includes returning an

address of program code of the module having its export information removed if the other module is authenticated.

24. The method of claim 15, wherein said removing step includes removing export information for a particular module specified by a user.

25. The method of claim 15, wherein said removing step includes removing information from an export table.

26. The method of claim 15, wherein said securely storing step includes obscuring removed export information.

27. The method of claim 15, further comprising:

in response to an attempt to invoke program code of a given module, verifying authenticity of the given module and blocking the attempt if the given module cannot be authenticated.

28. The method of claim 15, further comprising:

after allowing the attempt to proceed, providing for subsequent attempts by the other module to invoke the program code to directly invoke the program code.

29. A computer-readable medium having processor-executable instructions for performing the method of claim 15.

30. The method of claim 15, further comprising:

providing a downloadable set of processor-executable instructions for performing the method of claim 15.

31. A system for securing a program comprised of a plurality of interoperable components, the system comprising:

a module for extracting export information about a function of a first component

of the program that is callable by at least one other component of the program;

a module for securing the extracted export information;

a validation module for validating authenticity of a second component attempting to obtain export information to invoke the function of the first component; and

a security module for blocking the attempt to invoke the function of the first component if the second component cannot be authenticated.

32. The system of claim 31, wherein the validation module validates authenticity of the second component based on examining a digital signature of the second component.

33. The system of claim 31, further comprising:

a module for generating a signature for at least some components of the program, so as to enable authentication of said at least some components.

34. The system of claim 33, wherein the module for generating generates a selected one of an Authenticode signature and an MD5 message digest.

35. The system of claim 33, wherein the module for generating generates a hash of a module and encrypts the hash with a private key.

36. The system of claim 35, wherein the validation module decrypts the hash with a public key and compares the hash to a known value.

37. The system of claim 31, wherein the security module routes the attempt to invoke the function to the first module using the extracted export information if the second module is authenticated.

38. The system of claim 31, wherein the security module returns an address of the function of the first module if the second module is authenticated.

39. The system of claim 31, wherein the module for extracting removes an export table entry for the function of the first module.

40. The system of claim 31, wherein the module for securing creates a secure export table including the extracted export information.

41. The system of claim 40, wherein the secure export table is created without using a clear text name for the function of the first module.

42. The system of claim 40, wherein the module for securing obscures function names in the secure export table.

43. The system of claim 31, wherein the security module inserts executable code into the second module during initialization of the second module so as to direct an attempt by the second module to invoke the function of the first module to the security module.

44. The system of claim 43, wherein the security module modifies the executable code included in the second module if the second module is authenticated so as to enable the second module to directly invoke the function of the first module.

45. A method for securing an exported function of a program, the method comprising:

extracting export information about the exported function of the program;

securing the extracted export information;

intercepting an attempt to access the exported function by an importer;

authenticating the importer for determining whether to permit access to the exported function;

if the importer is authenticated, providing access to the exported function based on the secured extracted export information; and

otherwise, blocking access to the exported function.

46. The method of claim 45, wherein the importer comprises another module of the program.

47. The method of claim 45, wherein the importer comprises another program.

48. (Canceled)

49. The method of claim 45, wherein said authenticating step includes authenticating the importer based on a digital signature of the importer.

50. The method of claim 45, further comprising:

generating a digital signature for at least some executable modules of the program, so as to enable authentication of said at least some executable modules.

51. The method of claim 50, wherein said generating step includes generating a selected one of an Authenticode signature and an MD5 message digest.

52. The method of claim 50, wherein said authenticating step includes validating digital signature of the importer.

53. The method of claim 45, further comprising:

authenticating a program module including the exported function before providing access to the exported function.

54. The method of claim 45, wherein said providing step includes routing a call by the importer to the exported function.

55. The method of claim 45, wherein said providing step includes returning an address of the exported function to the importer.

56. The method of claim 45, wherein said extracting step includes removing an export table entry for the exported function.

57. The method of claim 45, wherein said securing step includes obscuring the exported function name.

58. The method of claim 45, wherein said securing step includes creating a secure export table based on the extracted export information.

59. A computer-readable medium having processor-executable instructions for performing the method of claim 45.

60. The method of claim 45, further comprising:

providing a downloadable set of processor-executable instructions for performing the method of claim 45.

9. EVIDENCE APPENDIX

This Appeal Brief is not accompanied by an evidence submission under §§ 1.130, 1.131, or 1.132.

10. RELATED PROCEEDINGS APPENDIX

Pursuant to Appellant's statement under Section 2, this Appeal Brief is not accompanied by any copies of decisions.